AoI-Based Temporal Attention Graph Neural Network for Popularity Prediction and Content Caching

Jianhang Zhu, *Graduate Student Member, IEEE*, Rongpeng Li, *Member, IEEE*, Guoru Ding, Chan Wang, Jianjun Wu, Zhifeng Zhao, *Member, IEEE*, and Honggang Zhang, *Senior Member, IEEE*

***Abstract*—Along with the fast development of network technology and the rapid growth of network equipment, the data throughput is sharply increasing. To handle the problem of backhaul bottleneck in cellular network and satisfy people's requirements about latency, the network architecture, like the information-centric network (ICN) intends to proactively keep limited popular content at the edge of network based on predicted results. Meanwhile, the interactions between the content (e.g., deep neural network models, Wikipedia-alike knowledge base) and users could be regarded as a dynamic bipartite graph. In this paper, to maximize the cache hit rate, we leverage an effective dynamic graph neural network (DGNN) to jointly learn the structural and temporal patterns embedded in the bipartite graph. Furthermore, in order to have deeper insights into the dynamics within the evolving graph, we propose an age of information (AoI) based attention mechanism to extract valuable historical information while avoiding the problem of message staleness. Combining this aforementioned prediction model, we also develop a cache selection algorithm to make caching decisions in accordance with the prediction results. Extensive results demonstrate that our model can obtain a higher prediction accuracy than other state-of-the-art schemes in two real-world datasets. The results of hit rate further verify the superiority of the caching policy based on our proposed model over other traditional ways.**

***Index Terms*—Content caching, popularity prediction, dynamic graph neural network, age of information.**

Manuscript received 11 March 2022; revised 30 September 2022; accepted 23 November 2022. Date of publication 9 December 2022; date of current version 11 April 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 61731002 and 62071425, in part by the Zhejiang Key Research and Development Plan under Grants 2022C01093, 2019C01002 and 2019C03131, in part by Huawei Cooperation Project, and in part by the Zhejiang Provincial Natural Science Foundation of China under Grant LY20F010016. The associate editor coordinating the review of this article and approving it for publication was N. Zhang. *(Corresponding author: Zhifeng Zhao.)*

Jianhang Zhu, Rongpeng Li, and Chan Wang are with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: zhujh20@zju.edu.cn; lirongpeng@zju.edu.cn; 0617464@zju.edu.cn).

Guoru Ding is with the College of Communications and Engineering, Army Engineering University of PLA, Nanjing 210007, China (e-mail: dr.guoru.ding@ieee.org).

Jianjun Wu is with the Wireless Communications Lab, Huawei Technologies Company Ltd., Shanghai 201206, China (e-mail: wujianjun@huawei.com).

Zhifeng Zhao is with the Research Institute of Intelligent Equipment, Zhejiang Lab, Hangzhou 311121, China, and also with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China (e-mail: zhaozf@zhejianglab.com).

Honggang Zhang is with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China, and also with the Editorial Office of Intelligent Computing, Zhejiang Lab, Hangzhou 311121, China (e-mail: honggangzhang@zju.edu.cn).

Digital Object Identifier 10.1109/TCCN.2022.3227920

I. Introduction

GIVEN the galloping number of users and mobile equipment [1], the amount of data sharply surges and the wireless access points at the network edge confront frequent congestion. Generally, besides video streaming, provisioning artificial intelligence (AI) and other dedicated network functions services are becoming the dominant factors that steer this explosion. Therefore, how to bring a better quality of user experiences (QoE) and quality of service (QoS) to users in the sharply growing data traffic under a constrained backhaul link is an intractable problem we have to face. Some resource-devouring approaches, such as higher frequency reuse, a larger scale of antennas or setting more bandwidth, can tackle this problem by increasing the capacity of cellular networks, but most of them fail to offer a durable solution in terms of scalability, costs and flexibility [2]. On the other hand, some studies [3], [4] point out that a tremendous data load comes from the repeated requests for a few same popular targets, especially the multimedia services at the edge, so does the scenario of AI service, in which the same AI samples or trained models may devote to numerous applications [5]. Therefore, aiming at dealing with these repetitive actions, some studies propose to store part of fashionable content (e.g., deep neural network models, videos) at the network edge, which can obviously alleviate the backhaul traffic burden caused by the data explosion and greatly reduce the transmission delay or other issues [6]. So there is a growing consensus that edge caching will play a prominent role in future communication systems and networks [7].

The demands for a more efficient and much simpler content distribution method have motivated the emergence of a new architecture called information-centric network (ICN) [8]. In contrast to the inefficient way like IP addressing, clients in ICN are able to directly access content pieces from the network only by its unique named data objects (NDO) [9]. As illustrated in Fig. 1, ICN can easily satisfy requests by any edge node if the node holds a copy with the exact NDO in its in-network storage [10]. Practically, due to the limited cache space [11], we are only able to recommend those content with distinguished cost performance to ICN's in-network storage. Ideally, we should proactively cache the most popular content. Most of the existing caching strategies always assume that the content popularity remains stable during a long period, while it actually varies over time [12]. For example, some

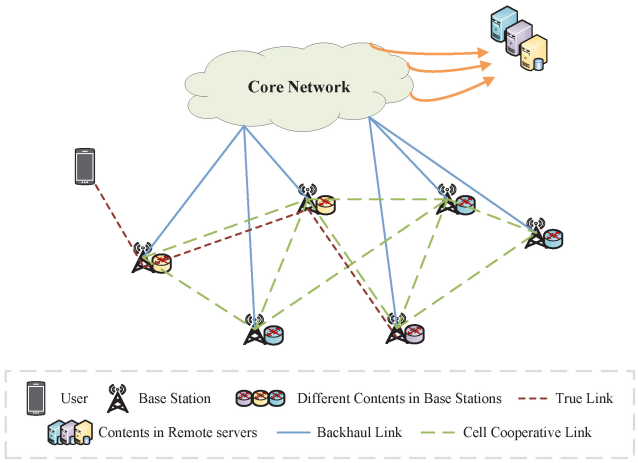


Fig. 1. Mobile edge caching in ICN.

traditional caching strategies like Least Recently Used (LRU) and Least Frequently Used (LFU) [13] are partial to extracting the superficial periodic law from historical information, and ignore the dynamic characteristics of requests themselves [14]. To enhance the caching performance, it becomes an incentive to lucubrate the dynamics to establish a popularity prediction model.

In addition to the dynamics of historical requests, we also believe a well-adopted popularity prediction model ought to predict from users' perspectives and excavate the structural pattern within data as well. In other words, it is inspiring to exploit the implications from users with similar preferences when speculating some inactive entities' predilections for comprehensive popularity anticipation. Recently, some researchers regard user-content pairs in recommendation system (RS) as a bipartite graph [15] and propose to utilize graph neural network (GNN), such as Graph Attention Network (GAT) [16], to dig out structured data. They demonstrate that even without dynamic information, GNN model also wins excellent performance in RS [17]. Inspired by these works [15], [16], [17], interactions between the content and users could also be regarded as a bipartite graph when we attempt to predict their popularity in ICN. However, it is also non-negligible that most real-life graphs are always evolving, and the ignorance of time-varying nature of the aforementioned approaches makes them still far from perfection [18].

In order to realize the learning of structure and dynamics simultaneously, the dynamic graph neural network (DGNN) has been proposed. As shown in Fig. 2, there are mainly two kinds of DGNN models (i.e., time sampling and event sampling). DGNN with time sampling obtains a series of sub-graphs by discrete sampling with time interval T. So the sampling period's granularity T is vital to the final results. If the time window is too tiny, as shown at the top of Fig. 2, it yields many snapshots without any new information and brings more redundancy to computation. To avoid this issue, DGNN with event sampling further tries to finish the building of dynamic graph from the insight of continuity by sampling at occurrence of events and recording the corresponding timestamps, as exemplified at the bottom of Fig. 2. But the efficacy of existing approaches still remains room to improve. In this article, we primarily leverage a modified continuous DGNN (CDGNN) model to learn the structural and temporal pattern in the dynamic bipartite graph of users and requested content. Specifically, we focus on discovering how to abstract temporal features and how many historical messages we should utilize while mining the dynamic features. To solve the above issues, we introduce age of information (AoI), a metric quantifying the freshness of data [19], to guide the selection of fresh information and a multi-head attention mechanism is employed to refine temporal characteristics. Both of them contribute to the generation of more precise representations for users' preference predictions. Afterwards, a strategy, which relies on the results of CDGNN, is proposed to guide the caching. Overall, the main contributions of this paper are as follows:

- To forecast users' preferences precisely, a CDGNN model is used to simultaneously excavate the structural and dynamic patterns of the bipartite graph for all users and we also design a caching policy on top of the CDGNN model.
- We develop an AoI-based temporal attention graph neural network (ATAGNN) method by innovatively introducing the AoI concept and incorporating the attention mechanism on top of the GNN model to effectively mine the temporal features in the dynamic graph. The combination of AoI and GNN may be of independent interest to the GNN community.
- Extensive simulation results also manifest the prediction accuracy of our method and confirm the superiority over other deep learning models. Meanwhile, we testify the strategy relies on our ATAGNN model's effectiveness within different cache spaces, and updating periods. The results of caching hit rate show our scheme significantly surpasses the performance of traditional policies like LRU and LFU.

The remainder of this paper is organized as follows: The related work and background are introduced in Section II. We present the system model in Section III. The details of our proposed AoI-based temporal graph neural network are delivered in Section IV. In Section V, we provide the numerical analysis and the results of prediction. Finally, the conclusion is summarized in Section VI. For convenience, we also list the major notations of this paper in Table I.

II. Related Work

Traditional caching policies, LRU and LFU, have obtained motivating results. Furthermore, one of their variants [20] proposes to combine age of information (AoI) with LRU to achieve cooperative caching between several cache-enabled edge servers, but due to the common limitation of LRU and LFU, it still fails to mine the dynamics under the superficial statistical law. To make a dynamic caching decision, utilizing the content popularity as a reference is a commonly-adopted way. Some innovative studies based on locally deployed popularity prediction algorithms have thrived in recent years [21], [22], [23], [24]. In particular, with the

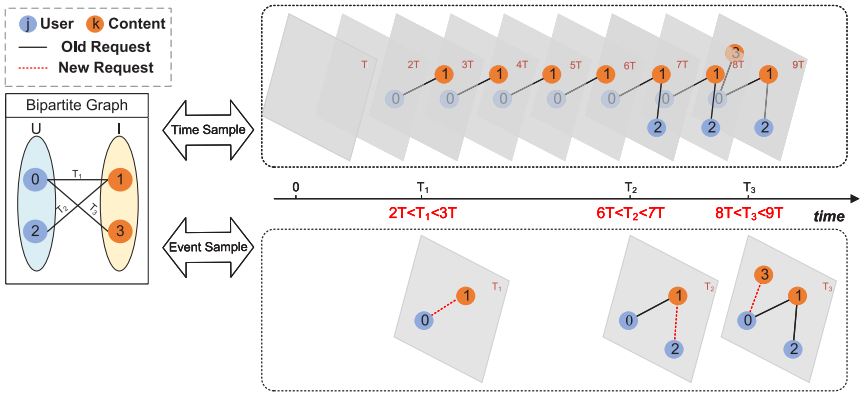


Fig. 2. An example of time sampling and event sampling to dynamic graph, in which the sampling period of time sampling is T and three events occurred at T_1, T_2 and T_3, and $2T < T_1 < 3T$, $6T < T_2 < 7T$ and $8T < T_3 < 9T$.

TABLE I
A SUMMARY OF MAJOR NOTATIONS USED IN THIS PAPER

Notation	Definition
u_j, i_k	The notations of user j and content k, for $j \in [0, J]$ and $k \in [0, K]$; J is the max number of users, K is the max number of content.
$\boldsymbol{v}_{u_j}, \boldsymbol{v}_{i_k}, \boldsymbol{e}_{jk}$	The features of user j, content k and their edge.
δ_p	The updating period of content.
$p_j^k(\delta_p)$	The actual possibility of content k requested by user j during the content updating period δ_p.
$\tilde{p}_j^k(\delta_p)$	The possibility of content k requested by user j during the content updating period δ_p that is calculated by our model.
$A^k(\delta_p)$	The total request behaviors about content k for all users during the updating period δ_p.
$\tilde{A}^k(\delta_p)$	The total request behaviors about content k for all users during the updating period δ_p that we predict.
P_I	The threshold value for determining whether a request will occur or not.
$\mathcal{C}(\delta_p)$	The set of top-C content that we predict to be cached.
$\mathrm{F}(\cdot)$	A multi-layer perceptron that is used to calculate the preference between users and content.
$\Delta_{jk}(t, n)$	The age of user j's n-th information.
Δ_{T_p}	The difference between the target timestamp T_p to predict and the most recent timestamp T_l of a request in history
T_u	The period that we adopt to update memory.

development of AI algorithms, deep learning plays a non-negligible role in the popularity prediction and caching tasks. For instance, [12] uses a feed-forward neural network (FNN) for estimating the caching threshold to assist the caching decision. Reference [25] proposes to use a recurrent neural network (RNN) to recommend popular content, but it is complicated to be generalized to a new dataset that has not been trained because of the limitations of RNN. What's worse, both of them fail to exploit the interdependency among all users or the structural pattern of data. In recent years, due to the excellent performance of extracting the structural pattern of a graph, some studies have succeeded in using graph neural network (GNN) to realize the popularity recommendation by regarding users and their requested items as the components of a bipartite graph [26]. In that regard, GAT is verified to effectively learn the structure-rich representations by aggregating interrelated vertexes in the recommendation graph with weights calculated by attention mechanism [27].

Although GNN, GAT and their variants have yielded excellent performance in representation learning and won remarkable achievements in recommendation tasks, they neglect the impact of dynamic features and still have room to improve. Therefore, GNN models with dynamics learning have been proposed to bridge the gap. For example, DySAT [28] and most DGNN models at the early stage achieve their dynamics extraction by sampling series snapshots from the evolving graph with equal time intervals. But the choice of sample granularity is a prominent part of model designing, for which an inappropriate granularity may result in the failure of yielding a snapshot with a new effective graph structure [29]. To avoid this obstacle, some continuous dynamic graph models (CDGNN), e.g., DyRep [30], propose to complete the graph computation with event sampling. Specifically, DyRep expresses a dynamic graph as the evolution of structural and node communication with a recurrent architecture. Moreover, some researchers inspired by the position encoding in Transformer [31] try to inject some information about interaction timestamps into the node. For instance, TGAT [18] uses the harmonic processing method to obtain a time coding function. Furthermore, TGN [32] intends to refine the temporal signals of historical interactions by adding a memory module to the TGAT and achieves superior performance in the above DGNN models. To some extent, the improvement stems from the extraction of users' short-term and long-term preferences with the memory module. However, during aggregating history, TGN, which calculates the mean value or keeps the latest records in its memory module, is relatively preliminary. A further refinement of short-term history is necessary.

As described, aforementioned models like TGN are becoming an important tool to realize the proactive caching, but still leave questioned on how to design a much more effective information aggregation method in CDGNN. Therefore, we introduce an attention mechanism to the TGN model so as to capture both structural and dynamic patterns. And inspired by the AoI in wireless sensor network (WSN) [19], a metric of information freshness to balance the huge data and the limited transmission capacity, we novelly introduce the concept into our neural network for selecting fresh messages adaptively.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A. System Model

1) Content Caching Model: In this paper, we consider a network that contains some edge servers that provide in-network storage, e.g., the base stations (BSs). In each edge

server, there are multiple users in its coverage area and these clients always request content aperiodically. To realize the goal of caching, we compute content popularity for the users within the edge server and download the most popular ones. The main purpose of our paper is to obtain a better hit rate with a well-performed popularity prediction model.

For such a user-content system with J users and K contents, the set of users can be indicated by $\mathcal{U} = \{u_0, u_1, \ldots, u_J\}$ and the set of available contents in the network are identified as $\mathcal{I} = \{i_0, i_1, \ldots, i_K\}$. For all the entities, we define their necessary raw information (e.g., the age of a user or the category of content) as the features: $\mathcal{V}_\mathcal{U} = \{\boldsymbol{v}_{u_0}, \boldsymbol{v}_{u_1}, \ldots, \boldsymbol{v}_{u_J}\}$ and $\mathcal{V}_\mathcal{I} = \{\boldsymbol{v}_{i_0}, \boldsymbol{v}_{i_1}, \ldots, \boldsymbol{v}_{i_K}\}$. $\boldsymbol{v}_{u_j}$ and $\boldsymbol{v}_{i_k}$, where $j \in [0, J]$ and $k \in [0, K]$, denote the original features of user j and content k, respectively.

To accomplish the selection of future popular content, we define the possibility of content k requested by user j during the content updating period δ_P as $p_j^k(\delta_p)$. In this paper, we adopt DGNN model to calculate the possibility, as $\tilde{p}_j^k(\delta_p)$. Generally, since users at different time have different preferences towards the same content, there emerges an implicit evolving relationship between users and the requested content. To finish the task of caching, we need an action indicator function to indicate whether an item will be requested or not. The total request behaviors about content k for all users $\mathcal{U}$ within the edge server during the updating period can be formed as:

$$A^k\left(\delta_p, p_j^k\right) = \sum_{j \in \mathcal{U}} \mathbf{1}\left(p_j^k(\delta_p) > P_I\right), \ \forall k \in \mathcal{I}, \tag{1}$$

where P_I is the threshold value for requesting and any potential interaction with a possibility that is larger than P_I will be regarded as the possible request and recorded with the function $\mathbf{1}(\cdot)$. Moreover, we define the predicted request actions toward content k during the updating period as $\tilde{A}^k(\delta_p, \tilde{p}^k)$.

Due to the limited storage space, we denote the maximum number of the edge server's capability as $C \geq 0$. After predicting and counting users' possible actions, we then make a popularity ranking list for content in $\mathcal{I}$ and rely on the set of top-C items $\mathcal{C}(\delta_p)$ to update the cache space. Given the cached set $\mathcal{C}(\delta_p)$, the hit rate of our scheme during the content updating period δ_p can be denoted as $h(\delta_p)$, can be calculated as:

$$h\left(\delta_p\right) = \frac{\sum_{k \in \mathcal{C}(\delta_p)} \tilde{A}^k\left(\delta_p, \tilde{p}_j^k\right)}{\sum_{k' \in \mathcal{I}} A^{k'}\left(\delta_p, p_j^k\right)}. \tag{2}$$

2) Dynamic Graph Model: Note that the choice of $\mathcal{C}(\delta_p)$ is the important part for the increase of caching hit rate, and it has a positive relationship with the possibility of request behavior that we need to estimate. Undoubtedly, an accurate prediction towards popularity is the key part of a caching strategy, but users and their interests are often subject to change over time, which increases its difficulty. To proactively cache content in advance, we can forecast the popularity based on the historical request information. Considering the existence of inactive users, it will be much more effective to learn from users' structural patterns. Thus, we can utilize the technique of GNN and view the whole system as an evolving graph.

We reformulate the aforementioned sets as the components of a bipartite graph $\mathcal{G}$, where users and content can be regarded as vertexes of the dynamic graph, and the interactions are naturally identified as the bipartite graph's evolving edges, $\mathcal{E} = \{\boldsymbol{e}_{01}, \boldsymbol{e}_{21}, \boldsymbol{e}_{03}, \ldots, \boldsymbol{e}_{jk}\}$, where $\boldsymbol{e}_{jk}$ represents the vector of interaction between user j and content k. As time goes by, correspondingly, new edges, as well as their participants, are added into the dynamic graph, as shown in the bottom of Fig. 2. For the sake of simplicity, we don't consider the choice between multiple content providers for the same content, and we also assume that all the different files have an equal size.

Furthermore, the occurrence of new requests can be deemed as the generation of edges, and for a dynamic graph, its timestamp should also be simultaneously recorded: $\mathcal{G} = \{(\boldsymbol{R}_{01}, T_1), (\boldsymbol{R}_{21}, T_2), (\boldsymbol{R}_{03}, T_3), \ldots, (\boldsymbol{R}_{jk}, T_N)\}$, where T_N denotes the occurrence timestamp of the edge, and $\boldsymbol{R}_{jk} = \{\boldsymbol{v}_{u_j}, \boldsymbol{v}_{i_k}, \boldsymbol{e}_{jk}\}$. We use $\mathbf{Inf}_{jk} = f(\boldsymbol{v}_{u_j}, \boldsymbol{v}_{i_k}, \boldsymbol{e}_{jk}) \in \mathbb{R}^d$ to demonstrate an interaction event, where $f(\cdot)$ is the concatenation function, to denote the message between user j and content k. Similarly, the initial embedding of a message from the perspective of content k is presented as $\mathbf{Inf}_{kj} = f(\boldsymbol{v}_{i_k}, \boldsymbol{v}_{u_j}, \boldsymbol{e}_{jk})$. For node u_j at time t, we denote the set of its request targets as: $\mathcal{N}(\boldsymbol{v}_{u_j}; t) = \{\boldsymbol{v}_{i_0}, \boldsymbol{v}_{i_1}, \ldots, \boldsymbol{v}_{i_M}\}$. Subsequently, a DGNN model can be employed to generate representations containing structural and dynamic patterns for further computation. Moreover, since the interactions we cope with are instantaneous, so we ignore the lasting time of an interaction and only focus on the occurrence of the request.

B. Temporal Graph Network Model for Popularity Prediction

We discuss how to predict the popularity after formulating the interaction between users and the requested content as a dynamic bipartite graph. Since the graph is not static, we refer to the temporal graph network [32] for discovering its underlying temporal relationship concurrently. The components of this model are defined as follows:

- The time-coding module, which is proposed in [18], is used to encode the timestamp of a request's built-up time. However, the interval between the latest request and the target one is much more meaningful than the absolute time points, so we use the difference Δ_t instead. To some extent, it achieves a preliminary extraction of temporal information. Consistent with TGAT, the function is defined as:

$$\Phi_{d_T}(\Delta_t) = \sqrt{\frac{1}{d_T}}[\cos(\omega_1 \Delta_t), \cos(\omega_2 \Delta_t), \ldots, \cos(\omega_d \Delta_t)]^T \tag{3}$$

  where $\Phi_{d_T}(\Delta_t) \in \mathbb{R}^{d_T}$, $\omega_1, \omega_2, \ldots, \omega_d$ are the parameters to be trained and d_T is the dimension number of the time embedding we want.
- The time-concatenating module combines the initial message embedding $\mathbf{Inf}_{jk}$ between users and content with the encoded time feature as the intact representation of an interaction, and the set of user j's all interactions is described as:

$$\mathbf{Msg}_j(t) = \left[\mathbf{Inf}_{j0} || \Phi_{d_T}(0), \ldots, \mathbf{Inf}_{jk} || \Phi_{d_T}(\Delta t_N)\right]^T \tag{4}$$

where the operator $||$ denotes a concatenation operation and $\mathbf{Msg}_j(t) \in \mathbb{R}^{N\times(d+d_T)}$ is the final input feature to the DGNN model.

- Inspired by TGN [32], a memory embedding module is also adopted to reinforce the dynamics refinement, i.e., the extraction of short-term and long-term interests of users. It aggregates the historical interactions stored in the message buffer to obtain an embedding with richer temporal information about short-term preference. Subsequently, it updates former memory with the embedding to inherit and renew the long-term one. Considering that the aggregation methods in TGN are elementary for extracting short-term temporal characteristic, we further propose to adopt an AoI-based attention mechanism for utilizing raw request information and we will talk about it later.
- Finally, an embedding module is adopted to extract the structural pattern within the graph and merge the time feature that we want to predict for further computation. The embedding module outputs the final representations, which contain both temporal and structural characteristics for all vertexes in the dynamic bipartite graph. Specifically, the final representation of user j and content k are denoted as $\mathbf{E}_j^u$ and $\mathbf{E}_k^i$. Based on these embedding representations, we can predict the preference of user j for content k as

$$\tilde{p}_j^k(\delta_p) = F\left(\mathbf{E}_j^u, \mathbf{E}_k^i\right) \tag{5}$$

where a multi-layer perceptron (MLP) is a good choice to achieve the function $F(\cdot)$. Besides, we extract the same number of negative samples to speed up the training. Finally, we use a binary cross entropy (BCE) loss function for optimizing the whole neural network:

$$\mathcal{L} = \frac{1}{n}\sum_{\gamma} -\left(y_\gamma \log\left(\tilde{p}_\gamma\right) + \left(1-y_\gamma\right)\log\left(1-\tilde{p}_\gamma\right)\right) \tag{6}$$

where y_γ is the label of the γ-th samples, $y_\gamma = 1$ if the γ-th is a positive sample and $y_\gamma = 0$ for negative samples. $\tilde{p}_\gamma = \tilde{p}_j^k$ is prediction result of the γ-th samples calculated by Eq. (5).

After estimating the user preference $\tilde{p}_j^k$, we can derive the request behaviors $\tilde{A}^k(\delta_p, \tilde{p}_j^k)$ and the caching policy from Algorithm 1 and Algorithm 2.

IV. AoI-Based Temporal Graph Neural Network

Although the future requests of a user are diverse, comparatively accurate predictions can still be speculated by aggregating and analyzing existing history information. To some extent, aggregating history information can extract the short-term interests of users. But as described above, the ways of aggregation in TGN are still far from perfection: 1) the method of always keeping the latest request may not have much negative impact on the trained nodes, but it may lead to some errors when a rookie node joins into the graph due to the lack of historical references. 2) the method of averaging ignores the fact that requests at different time have distinct influences on future behavior, and some are so obsolete that they may bring adverse effects. Thus, we propose an AoI-based attention mechanism to ameliorate the model.

A. Attention Mechanism for Temporal Pattern Extraction

As demonstrated in TGN, the more recent and repeated actions usually have greater impacts on the future interests prediction, we can calculate the degrees of correlation between previous interactions and the latest one with a self-attention mechanism [31], which are used as the indications of influence later. Hence, we name this model TGN-A for short. Notably, the weighted summation of all the chosen messages is the aggregated feature to update the memory, as shown in Fig. 3. The aggregated message of user j is calculated as follows:

$$\begin{aligned}\boldsymbol{h}_j(t) &= ATT\left(\mathbf{Msg}_j(t), \left[\mathbf{Msg}_j(t)\right]_0\right)\\ &= \sigma\left(\sum_{m\in N}\alpha_{jm}V_{jm}\right) \in \mathbb{R}^{d_h}\end{aligned} \tag{7}$$

where $\boldsymbol{V}_{jm} = [\mathbf{Msg}_j(t)]_m \boldsymbol{W}_V$, $\mathbf{W}_V \in \mathbb{R}^{(d+d_T)\times d_h}$ is the weight matrix that needs to be trained. α_{jm} is the attention coefficient, indicating the importance between the m-th history message and the most recent one, and α_{jm} can be specifically presented by:

$$\begin{aligned}\alpha_{jm} &= \mathrm{softmax}\left(\left[\mathbf{Msg}_j(t)\right]_m, \left[\mathbf{Msg}_j(t)\right]_0\right)\\ &= \frac{\exp\left(\left(\left[\mathbf{Msg}_j(t)\right]_0 \boldsymbol{W}_Q\right)\left(\left[\mathbf{Msg}_j(t)\right]_m \boldsymbol{W}_K\right)\right)}{\sum_{n=1}^{N}\exp\left(\left(\left[\mathbf{Msg}_j(t)\right]_0 \boldsymbol{W}_Q\right)\left(\left[\mathbf{Msg}_j(t)\right]_n \boldsymbol{W}_K\right)\right)}\end{aligned} \tag{8}$$

where the $\mathbf{W}_Q$, $\boldsymbol{W}_K \in \mathbb{R}^{(d+d_T)\times d_h}$ are the weights allocated to mix the request features with encoded time for producing the integrated features from different perspectives. Besides, $\mathrm{softmax}(\cdot)$ is a function that aims at normalizing the obtained coefficients and emphasizing the weights of important elements. $[\mathbf{Msg}_j(t)]_m$ or $[\mathbf{Msg}_j(t)]_n$ for $m, n = 1, \ldots, N-1$ are the m-th or n-th message in $\mathbf{Msg}_j(t)$ and $[\mathbf{Msg}_j(t)]_0$ is the latest one.

It is worth noting that, instead of considering the effect of all historical requests, inspired by GraphSAGE [33] and TGN [32], we only sample the most recent N historical messages for aggregating. As for those only with M interactions, where $M < N$, we pad their history behavior set $\mathbf{Msg}_j(t)$ with mask operation, as done in Transformer [31].

Besides, in view of the excessive smoothing effect incurred by increasing the number of network layers for information transmission and aggregation of nodes, we introduce the "skip-connection" derived from ResNet [34] to combine the obtained representation with the base message $[\mathbf{Msg}_j(t)]_0$ by a feed-forward network $FN(\cdot)$. We believe it will also improve the overall performance of the model by capturing non-linear interactions between the features:

$$\begin{aligned}\overline{\mathbf{Msg}_j(t)} &= \boldsymbol{h}'_j(t) = FN\left(\boldsymbol{h}_j(t)||R_{j0}\right)\\ &= \mathrm{ReLU}\left(\left(\boldsymbol{h}_j(t)||R_{j0}\right)\boldsymbol{W}_0 + \boldsymbol{b}_0\right)\boldsymbol{W}_1 + \boldsymbol{b}_1\end{aligned} \tag{9}$$

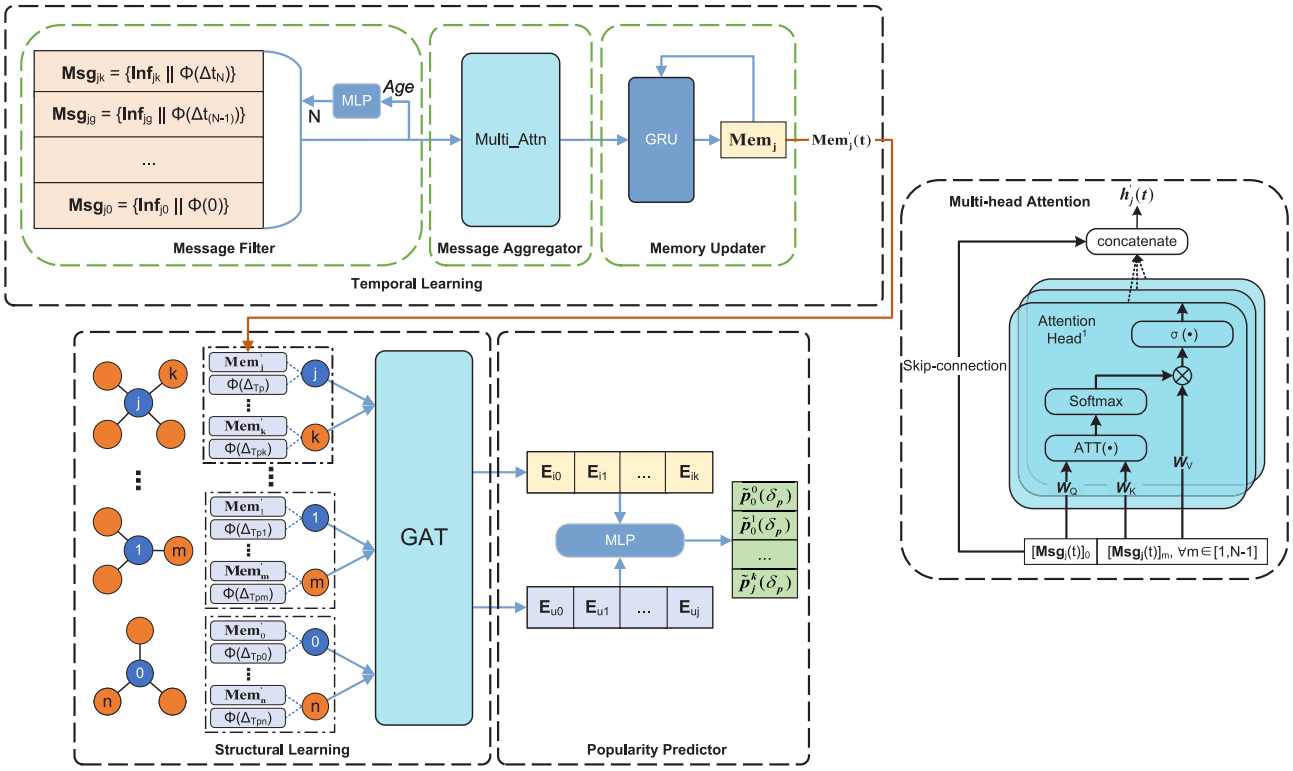


Fig. 3. The illustration of AoI-based temporal attention graph neural network and the multi-head attention mechanism.

where $\boldsymbol{h}'_j(t) \in \mathbb{R}^d$ is the final aggregated vector representing the time-aware embedding at time t, and can also be denoted as $\overline{\mathbf{Msg}_j(t)}$. And $\boldsymbol{W}_0 \in \mathbb{R}^{(d_h+d)\times d_0}$, $\boldsymbol{W}_1 \in \mathbb{R}^{d_0 \times d}$, $\boldsymbol{b}_0 \in \mathbb{R}^{d_0}$, $\boldsymbol{b}_1 \in \mathbb{R}^d$ are weight parameters to be trained in the neural network.

Empirically, [31] suggests that a multi-head attention may avoid the instability of the training process and promotes the performance of self-attention. We also extend the adopted attention to the multi-head setting, as presented on the right side of Fig. 3. We conduct an attention mechanism with L heads and record each head's output for user j and content k with Eq. (7) and Eq. (8) as: $\boldsymbol{h}_j^{(l)}(t) = ATT^{(l)}([\mathbf{Msg}_j(t)]^{(l)}, [\mathbf{Msg}_j(t)]_0^{(l)}, \boldsymbol{V}^{(l)}), l = 1, 2, \ldots, L$, and the weights values in different heads are various. Actually, it is a process that several single-head attentions carried out independently in parallel to gain deeper insights from different observation angles. Consequently, we modify Eq. (9) as:

$$\overline{\mathbf{Msg}_j(t)} = \boldsymbol{h}'_j(t) = FN\left(\boldsymbol{h}_j^{(1)}(t)||\boldsymbol{h}_j^{(2)}(t)||\cdots\boldsymbol{h}_j^{(L)}(t)||R_{j0}\right) \tag{10}$$

In our experiment, we find the final attention architecture with three heads ($L = 3$) will lead to a satisfactory result.

After extracting short-term interests with multi-head attention, we choose the Gated Recurrent Unit (GRU) [35] to finish the memory updating. The vector of new memory for user j, $\mathbf{Mem}'_j$, derived from the memory updater where update the former memory $\mathbf{Mem}_j$ with the aggregated message $\boldsymbol{h}'_j(t)$, and it contains all the history that we have chosen in the aggregating module. To some extent, it also captures the long-term interest within the request data:

$$\begin{aligned}\mathbf{Mem}'_j &= GRU\left(\mathbf{Mem}_j, \overline{\mathbf{Msg}_j(t)}\right)\\ &= GRU\left(\mathbf{Mem}_j, \boldsymbol{h}'_j(t)\right)\\ &= \boldsymbol{Z}_t \cdot \boldsymbol{H}_t + (1 - \boldsymbol{Z}_t) \cdot \mathbf{Mem}_j\end{aligned} \tag{11}$$

where $\boldsymbol{Z}_t$ is the update gate which decides the proportion of information that need to be inherited from the last hidden state, and $\boldsymbol{H}_t$ is the hidden state which is produced by ignoring some previous state $\mathbf{Mem}_j$ and resetting the current input $\overline{\mathbf{Msg}_j(t)}$ or $\boldsymbol{h}'_j(t)$ by a reset gate F_t. They are conducted as follows:

$$\begin{aligned}\boldsymbol{Z}_t &= \sigma\left(\boldsymbol{h}'_j(t)\,\boldsymbol{W}_{hZ} + \mathbf{Mem}_j\,\boldsymbol{W}_{MZ} + \boldsymbol{b}_Z\right)\\ \boldsymbol{F}_t &= \sigma\left(\boldsymbol{h}'_j(t)\,\boldsymbol{W}_{hF} + \mathbf{Mem}_j\,\boldsymbol{W}_{MF} + \boldsymbol{b}_F\right)\\ \boldsymbol{H}_t &= \tanh\left(\boldsymbol{h}'_j(t)\,\boldsymbol{W}_{hH} + \left(\boldsymbol{F}_t \cdot \mathbf{Mem}_j\right)\boldsymbol{W}_{MH} + \boldsymbol{b}_H\right)\end{aligned} \tag{12}$$

where $\boldsymbol{W}_{hZ}$, $\boldsymbol{W}_{hF}$, $\boldsymbol{W}_{hH}$, $\boldsymbol{W}_{MZ}$, $\boldsymbol{W}_{MF}$, $\boldsymbol{W}_{MH}$ are the weights of the recurrent neural networks, $\boldsymbol{b}_Z$, $\boldsymbol{b}_F$, $\boldsymbol{b}_H$ are their bias values. The activation function $\sigma(\cdot)$ is used to limit the result within 0 and 1. Due to the process of forgetting and updating, we obtain a new feature that contains user's long-term interests as well as the short-term ones. Similarly, a Long Short-Term Memory (LSTM) or other RNN architectures have been proven that they also have a similar gain in [18].

B. AoI-Based Attention Mechanism

Faced with massive historical information, one of the essential issues lies on deciding the amount of information that

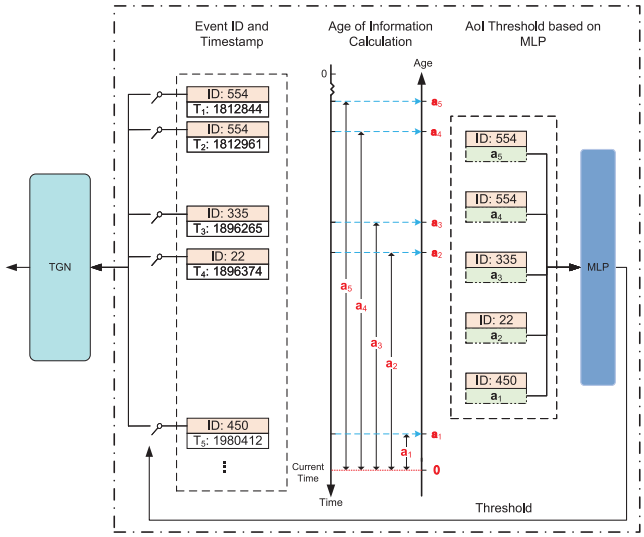


Fig. 4. An example of age of information for a user's request in Wikipedia dataset.

deserves to be aggregated. If the chosen requests occurred too long before, it may have a negative impact on the future, let alone using all precious requests, leading to an unnecessary increment in computing cost. Even we have restricted the size of aggregated neighborhoods, the selected information may still be non-positive to our prediction. To optimize the result, we introduce the concept of AoI, which is a commonly-adopted metric of information freshness in WSN. Inspired by the application of filtering fresh data in WSN, we tend to pass AoI to the TGN with attention (TGN-A) model for excluding the information with an age that is too stale to be positive for the final prediction. For simplicity of representation, we denote the TGN with AoI-based attention for temporal learning as ATAGNN.

As introduced in [36], the age of information is usually defined as:

$$\Delta_{jk}(t,n) = t - B_{jk}(n) \tag{13}$$

where $B_{jk}(n)$ represents the birth time of the n-th request between user j and content k, and t denotes the current time or the newest time point of the messages. Hence, we can easily figure out an age for each request. In fact, since our training is based on batches, we regard the latest moment in a batch as the independent variable. But unlike the traditional way of calculating AoI, we regard the requests between two nodes at various time points as different interactions to avoid the influence reductions of repeated actions in the follow-up calculation.

As shown in Fig. 4, we extract several request records of one user that contain timestamp information and calculate their respective ages with Eq. (13). Then, we concatenate all the age data as an N-length vector a_j, and pass it to a two-layer MLP module to figure out a threshold, so as to finish the adaptive selecting task of the worthy information for each vertex.

$$\begin{aligned} \boldsymbol{h}_{th} &= MLP(a_j) = \mathrm{ReLu}\left(a_j \boldsymbol{W}^1 + \boldsymbol{b}^1\right) \\ \mathrm{thre}_t &= MLP(\boldsymbol{h}_{th}) = \mathrm{ReLu}\left(\boldsymbol{h}_{th} \boldsymbol{W}^2 + \boldsymbol{b}^2\right) \end{aligned} \tag{14}$$

where $a_j \in \mathbb{R}^N$, $\boldsymbol{h}_{th} \in \mathbb{R}^{d_{th}}$ and $\mathrm{thre}_t \in \mathbb{R}$. Besides, $\boldsymbol{W}^1 \in \mathbb{R}^{N \times d_{th}}$ and $\boldsymbol{b}^1 \in \mathbb{R}^{d_{th}}$, $\boldsymbol{W}^2 \in \mathbb{R}^{d_t h}$ and $\boldsymbol{b}^2 \in \mathbb{R}^1$ are the training parameters of the 2-layer perceptron. We believe this module can simply fit the general behavior based on the age of requests and offer a threshold for determining whether the information deserves to be taken into consideration or not. And we discuss the details of the feasibility in the Appendix.

However, the fitting ability of an MLP is relatively preliminary, which will bring certain deviations inevitably. In order to decrease the potential impact of such deviations, we also adopt a "soft method" to promote the performance:

$$t'^n{}_{jk} = t^n_{jk} \cdot \sigma\big(100 * \big(\Delta_{jk}(t,n) - \mathrm{thre}_t\big)\big) \tag{15}$$

where t^n_{jk} is the raw timestamps of the n-th interaction between vertexes j and k in the dynamic graph $\mathcal{G}$, and $t'^n{}_{jk}$ is the one we actually use in subsequent calculation, thre_t is the threshold we obtain from the MLP. After calculating Eq. (15), we mask all the requests with an age that is lower than the threshold time, while redistributing a little greater timestamp to those that are close to the threshold. Then we execute the multi-head attention module with the new set of valuable messages, as mentioned before.

C. Future and Structural Patterns Embedding

As the final module in our model, we choose a GAT to accomplish the structure's deeper extraction and the generation of unique embedding representations for the participants of an interaction we want to predict. In order to map the future information, rather than the commonly adopted $\mathrm{LeakyReLu}(\cdot)$ in GAT [16], we promote the learning performance by adding the encoded target timestamps into the memory vectors and adopting a linear transformation with a dot-product:

$$\alpha^g_{jk} = \frac{\exp\left(\tilde{\boldsymbol{h}}_k \boldsymbol{W}^g_Q\right)^T \left(\tilde{\boldsymbol{h}}_j \boldsymbol{W}^g_K\right)}{\sum_{m \in \mathcal{N}(v_j;t)} \exp\left(\tilde{\boldsymbol{h}}_m \boldsymbol{W}^g_Q\right)^T \left(\tilde{\boldsymbol{h}}_j \boldsymbol{W}^g_K\right)} \tag{16}$$

where $\boldsymbol{W}^g_K$ and $\boldsymbol{W}^g_Q$ are the weight parameters that we employ to capture the relationship between time encoding and the output of temporal leaning $\mathbf{Mem}'_j$. Besides, the $\tilde{\boldsymbol{h}}_j = [\mathbf{Mem}'_j || \Phi_{d_T}(\Delta_{T_p})]$, and $\Delta_{T_p} = T_p - T_l$ is the difference between the target timestamp T_p to predict and the most recent timestamp T_l of a request in history from user j, where $T_p \geq T_l$. As we presented before, the more attention heads, the better structural representations will be extracted. Therefore, we also encapsulate a multi-head attention mechanism into this module.

We desire to generate reliable representations for the potential users and content by the above model, and calculate their correlation degree as a reference for judging users' coming behavior. Finally, we summarize the above algorithm in Algorithm 1. Meanwhile, we provide a comprehensive graphical illustration of our AoI-based temporal attention module for user j's temporal learning in the upper left part of Fig. 3.

V. Simulation Results and Numerical Analysis

In this section, we evaluate the performance of the models we mentioned above based on two real-world datasets:

Algorithm 1 The Preference Prediction Algorithm With AoI-Based Temporal Attention GNN

Input: Dynamic request dataset;

Output: The presentations of users E_j^u and content E_k^i. And predicate the preference between u_j and i_k p

Initialize the parameters for the whole network;
Initialize the memory buffer with zeros and message buffer.
Restore the graph information (**Inf** $\leftarrow$ all messages) and divide it into several mini batches;
for each $\mathrm{batch}(\boldsymbol{v}_{\mathrm{u_j}}, \boldsymbol{v}_{\mathrm{i_k}}, \boldsymbol{e}_{\mathrm{ui}}, \mathrm{t}) \in$ training dataset **do**
  $\dot{n} \leftarrow$ Sample negatives;
  Calculate the age $\Delta_{jk}(t, n)$ and the threshold thre_t;
  Filter and concatenate the valuable messages $\mathbf{Msg}(t)$;
  Aggregate with multi-head attention mechanism in Eq. (7), (10) and obtain $\overline{\mathbf{Msg}_j(t)}$;
  Update features $\mathbf{Mem}_j$ in memory buffer with GRU in Eq. (11);
  Encode the time difference Δ_{T_p} with Eq. (3) for all nodes;
  Concatenate the encrypted time feature with $\mathbf{Mem}_j$, and obtained the new feature $\tilde{\boldsymbol{h}}_j$ as the input;
  Obtain $\boldsymbol{E}_j^u(T_p)$ and $\boldsymbol{E}_k^i(T_p)$ through the modified *GAT*;
  Predict the correlation degree between users and content with Eq. (5);
  Optimize this network with $\mathrm{BCELoss}(\cdot)$;
end for

Algorithm 2 The Caching Algorithm With Neural Network

Initialize and load the whole neural network and the memory buffer;
for each $\mathrm{hour} \in [1, 24]$ **do**
  if $\mathrm{hour}\%\mathrm{T_u}! = 0$ **then**
    for $T_p = 0;\ T_p < 3,600;\ T_p + \delta_p$ **do**
      Load the trained memory;
      Obtain $\boldsymbol{E}_j^u(\delta_p)$ and $\boldsymbol{E}_k^i(\delta_p)$ with the neural network;
      Obtain the correlation degrees $\tilde{p}_j^k(\delta_p)$ between users and content;
      Count the possible activity with Eq. (1);
      Clear the memory buffer;
    end for
  else
    Update the trained memory with real data;
  end if
  Add up all the $\tilde{A}^k(\delta_p)$ and sort them with a list;
  Cache the top-C content according to the cache space.
end for

Wikipedia and MOOC. We also make a comparison between our models and four state-of-the-art methods designed for representation learning in temporal networks, including RNN [25], DyRep [30], TGAT [18] and TGN [32]. Besides, an experiment of cache hit rate in ICN between our prediction-based caching approach and the traditional policies (e.g., LRU and LFU) is also conducted.

A. Simulation Description

Wikipedia Dataset: It is a public dataset that records the Wikipedia pages edited by users on Wikipedia within 30 days. The number of entries and users involved is 9,227. There are more than 15,000 interactions, which represent the number of edges in the bipartite graph. Besides, their interactions are time-stamped. Finally, we perform a 70%-15%-15% chronological split for training, validation and testing.

MOOC Dataset: It is also a public dataset that records the history of actions done by students on a MOOC online course within one month, e.g., watching a video, or submitting an answer. We select 5,763 users and 56 contents as the nodes of the dynamic graph, which also consists of 175,856 time-stamped interactions. Due to the larger amount of interactions, we perform a 60%-20%-20% chronological split for training, validation and testing.

Evaluation Tasks and Training Configuration: One of the superiority of our model is that it can be easily generalized to the new data. Thus, we verify our model's performance in two types of tasks, i.e., transductive task and inductive task. In the transductive task, we evaluate model's ability of predicting the temporal links for those nodes that have been observed in the training phase, which means that the validation set and test set only contain those nodes that have appeared in the training set. In contrast, the target of inductive task is to inspect the model's talent in representing the nodes that have never been trained, so the nodes and interactions that we want to predict in validation and test are potentially different from the data in training phase. We also set a max number of aggregating for TGN-A and our ATAGNN model. To demonstrate the influence of AoI, we have a comparison between our model and TGN-A with a fixed age threshold (i.e., 60s, 600s, 3,600s.), and the max number of neighbors that we want to aggregate is 5. Simulation results in Table II demonstrates that our proposed model could yield superior performance in both tasks, especially in the sparse MOOC dataset.

Before the training, we sample an equal amount of negative interactions to the positive node pairs, regarding the prediction as a binary classifications problem, thus a BCELoss function is chosen. We adopt an Adam optimizer with a learning rate of 0.0001, and a batch size of 200 for all examinations. Finally, we adopt *Area Under the ROC Curve* (AUC) and *Average Precision* (AP) [29] as the metrics to indicate the performance.

Caching Policy Setting: When we deploy our DGNN model to produce the caching policy, unlike in the validation and test exams, we have little prior knowledge about the users, content and the possible timestamps of events. Inspired by LFU, we narrow the prediction scope by only choosing those entities that have been observed in the last one or two hours, for which the overall number of candidates is too large. We also design a caching algorithm that relies entirely on the results from the aforementioned model. The main idea of our policy is shown in Algorithm 2. It caches the contents from all candidate items by counting the possible accessible actions with

TABLE II
THE RESULTS OF TRANSDUCTIVE LEARNING AND INDUCTIVE LEARNING TASKS FOR PREDICTING FUTURE LINKS OF NODES. TGN-L, TGN-M AND TGN-A ARE THE TGN MODELS THAT KEEP THE LATEST MESSAGE, AVERAGING ALL MESSAGES AND USING AN ATTENTION MECHANISM FOR A FIXED NUMBER OF MESSAGES, RESPECTIVELY. 5N, 10N AND 15N MEAN THE NUMBER OF NEIGHBORS THAT THE ATTENTION MECHANISM AGGREGATES. THE BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN **BOLD** FONT AND THE BEST RESULTS IN THE BASELINE ARE HIGHLIGHTED IN underline

Dataset		Wikipedia				MOOC			
Metric		Old AUC	Old AP	New AUC	New AP	Old AUC	Old AP	New AUC	New AP
Baseline	RNN	76.476	77.100	-	-	70.394	70.461	-	-
	DyRep	93.823	94.313	91.795	92.668	87.498	83.591	87.201	83.493
	TGAT	95.391	95.710	93.241	93.810	74.413	69.932	73.211	69.282
	TGN-L	98.406	98.470	97.700	97.805	92.026	89.855	92.447	90.494
	TGN-M	98.342	98.426	97.741	97.864	90.951	88.423	92.271	90.319
TGN-A	5n	98.481	98.559	97.806	97.911	93.091	90.944	92.994	91.014
	10n	98.509	98.600	97.923	98.044	93.088	91.078	92.737	90.783
	15n	98.508	98.589	97.909	98.029	93.384	91.280	93.103	91.090
ATAGNN without Eq. (15)	5n	98.514	98.591	97.925	98.035	93.183	91.150	93.109	91.111
	10n	98.530	98.605	97.883	98.013	93.554	91.603	93.310	91.421
	15n	98.510	98.594	97.955	98.074	93.541	91.516	**93.424**	**91.478**
ATAGNN with Eq. (15)	5n	**98.544**	98.625	97.915	98.026	93.362	91.396	93.302	91.394
	10n	98.526	98.610	97.941	**98.066**	**93.577**	**91.635**	93.330	91.376
	15n	98.539	**98.632**	**97.957**	98.061	93.568	91.572	93.269	91.300

Eq. (1) as well as the results of our model and generating "fake requests" to download the popular content in advance. Moreover, if we update the memory with those fake requests, they may mislead our subsequent prediction. Thus, we need to update the memory with real interactions periodically, and call it the memory update period, T_u.

We compare our popularity-based scheme with LRU and LFU. LRU updates the caching by replacing the content that has not been requested for a longest time, while LFU always tries to keep those that have been most requested. Moreover, to show the superiority of our ATAGNN over other models in caching, we also compare it with the algorithm based on the best state-of-the-art model, TGN-L.

Cache-Hit Efficiency: Due to the lack of information for predicting with LRU and LFU in the first hour, the comparison with the traditional schemes is completed based on the testing data of MOOC and Wikipedia within 23 hours and the cache size is set as 15. There are 55 content in MOOC and more than 500 items in Wikipedia with around 2,000 users involved in our caching test.

Besides, the results of hit rate with various caching spaces are evaluated as well. On the other hand, in all the aforementioned tests, the default models' updating period for memory buffer is 24 hours, i.e., it will generate all results at once and never update the model's memory in our simulation. Therefore, we also carry out ablation studies to demonstrate the effectiveness of different memory update periods (e.g., $T_u = 24, 12, 4, 1$).

Moreover, we calculate users' preference to the content every 6-second with flexible thresholds in different datasets and list the ranking of the popularity for each hour. The tests also include transductive and inductive tasks.

B. Results Analysis

Prediction result: Table II presents the prediction accuracy. It can be observed that compared with some latest models based on DGNN or RNN, an attention mechanism based TGN model can provide deeper insights into the temporal information, which leads to better results in both inductive and transductive tasks, even without AoI. Especially in the MOOC dataset, the average precision will reach an improvement of about 2% after being integrated with an attention mechanism to aggregate historical interactions. Meanwhile, Table III demonstrates the positive impact of AoI when the age threshold is 3,600s, and it also clearly presents that our adaptive model can still win superior performance.

The results of TGN-L and TGN-M, as well as our models shown in Table II also prove that the number of aggregated information is an important factor that affects the results. Besides, it is apparent that after taking the AoI as a reference for choosing information, even without the Eq. (15), our model is able to reach further improvement, since it effectively excludes some stale information for certain vertexes. Furthermore, due to the introduction of Eq. (15), the model can slightly alleviate the error caused by the calculation of the threshold in the two-layer MLP, especially in transductive tasks.

Fig. 5 presents the convergence performance between our proposed models (i.e., TGN-A and ATAGNN) and some major baselines (i.e., TGN-L, TGN-M, TGAT and DyRep). Obviously, when convergence is achieved, our proposed models have lower loss values. In addition, we can also find that the loss value decreases faster in our model. Moreover, combined with the final results, as validated in Table II, although the final convergence loss values are similar with TGN-L and

TABLE III
THE RESULT OF TGN-A WITH FIXED AGE THRESHOLD (I.E., 60S, 600S, 3,600S), THE ORIGIN MODEL WITHOUT CONSIDERING THE AGE AND THE ATAGNN MODEL WITH MLP FOR ADAPTIVELY CHOOSING THRESHOLD. THE BEST RESULTS IN EACH COLUMN ARE HIGHLIGHTED IN **BOLD** FONT

Datasets		MOOC			
Metric		Old AUC	Old AP	New AUC	New AP
TGN-A with Age	60	92.363	90.191	92.234	90.004
	600	92.695	90.612	92.686	90.696
	3600	93.088	91.026	92.792	90.641
TGN-A without Age		93.091	90.944	92.994	91.014
ATAGNN		**93.362**	**91.396**	**93.302**	**91.394**

TABLE IV
COMPLEXITY FOR EACH MODEL AND TRAINING RUN-TIME IN MOOC

Model	Complexity	Training Time	Note
DyRep	$O(2Md)$	27.608s	-
TGAT	$O((kM_0d)^l)$	177.140s	$k=2, l=2$
TGN-L	$O((kM_0d)^l)$	23.740s	$k=2, l=1$
TGN-M	$O(Nd+(kM_0d)^l)$	75.380s	$k=2, l=1$
TGN-A	$O(k_0(N^2d+Nd^2)+(kM_0d)^l)$	187.088s	$k=2, l=1, k_0=2$
ATAGNN	$O(k_0(N^2d+Nd^2)+(kM_0d)^l)$	187.683s	$k=2, l=1, k_0=2$

TGN-M, our models avoid the over-fitting problem in these two original TGN models, which indicates a much stronger adaptability of our models.

One of the major concerns comes from the computational complexity and the time cost in training and practical application. As a result, we analyze the time complexity of each algorithm based on GNN model, and summarize it in Table IV, where d is the number of feature dimensions. M is the total number of the target's neighbors. N and M_0 represent the number of samples adopted in message aggregator module and the embedding module, respectively. k and k_0 are the numbers of heads that we adopt in the multi-head attention, and l is the layer number of GAT module. Notably, the memory module distinguishes TGN from TGAT, but the complexity of memory module in TGN-L is approximately equal to a constant, so TGAT and TGN-L have similar computational complexity. However, consistent with the settings in [32], we set the $k=2$, $l=2$ in TGAT, while TGN models can achieve their best effect at $k=2$, $l=1$, making the training time of TGAT steeply increased. The multi-head self-attention layer is adopted in our proposed model's aggregator module, and its complexity $O(N^2d+Nd^2)$ has a huge gap between TGN-L's O (1) and TGN-M's O (Nd), which also results in a trundling training process. But it is undeniable that it also leads to more significant performance improvement in the more sparse dataset MOOC.

Fig. 6 and Fig. 7 investigate the caching performance sensitiveness of our method. Fig. 6 reveals the evolution of the caching strategy based on our model and the best state-of-the-art algorithm TGN-L as well as the traditional caching algorithms' hit rate within 23 hours with a cache space of 15. Thanks to the sufficient aggregation of historical information, ATAGNN based caching is able to keep superior for a long time, especially in Wikipedia. Besides, the smooth

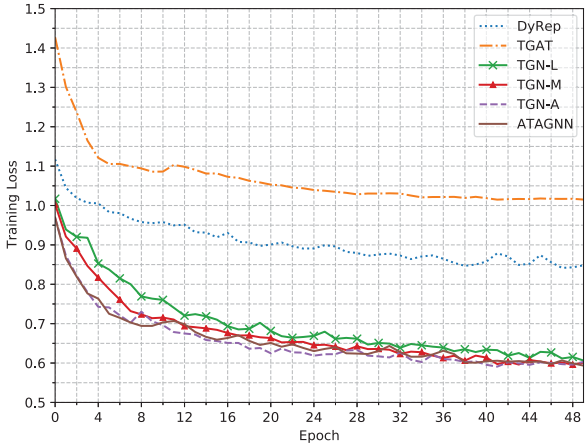


Fig. 5. Performance of training loss with different GNN models in MOOC.

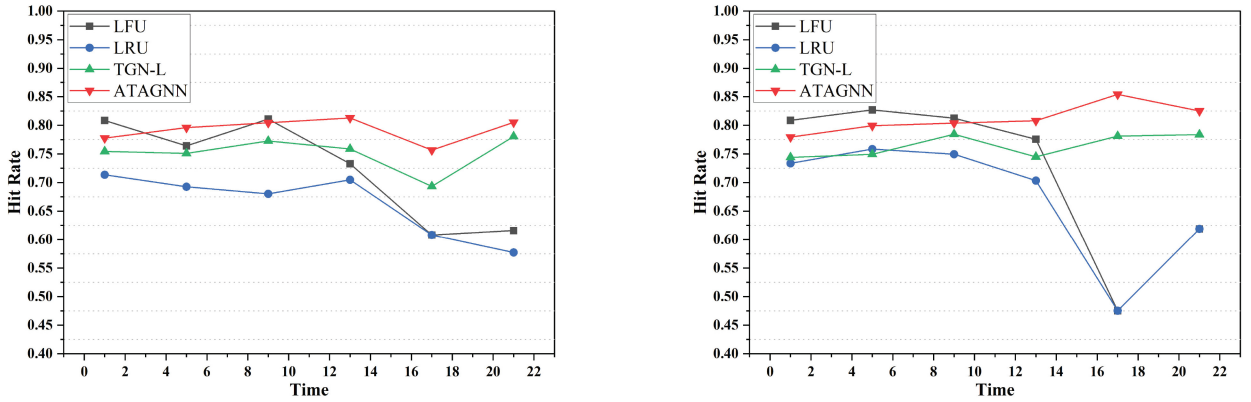
TABLE V
THE PERFORMANCE OF MODELS IN MOOC DATASET WITH TWO DIFFERENT MEMORY UPDATE PERIODS, (I.E., 24 HOURS AND 1 HOUR), AND DIFFERENT USER SETS

Dataset		Transductive				Inductive			
Model		TGN-L		ATAGNN		TGN-L		ATAGNN	
Cache Size		24	1	24	1	24	1	24	1
Past users	10	47.106	46.527	49.074	48.848	50.555	49.374	52.870	53.182
	15	61.944	62.097	65.190	65.860	63.673	62.290	67.598	67.025
	20	75.057	74.486	79.170	77.979	76.384	72.942	81.114	79.947
Future users	10	53.111	53.020	51.620	52.234	55.311	53.983	56.280	57.807
	15	68.755	68.228	69.404	70.314	70.040	67.640	71.896	75.235
	20	81.548	80.788	83.284	84.685	80.078	77.653	84.684	87.397

performance within the MOOC dataset also implies the stability of our models over traditional methods in caching. Fig. 7 shows the average hit rate of our caching policy with different maximum cache sizes (i.e., 5, 10, 15, 20). It can be observed that our model can always provide a high-confidence prediction result of the popularity for caching and greatly improve the hit rate in most cases. Moreover, as the cache space increases, the effect will be further improved. As for the inductive task, even the prediction result is not as good as the transductive one, our model can always keep its leading role.

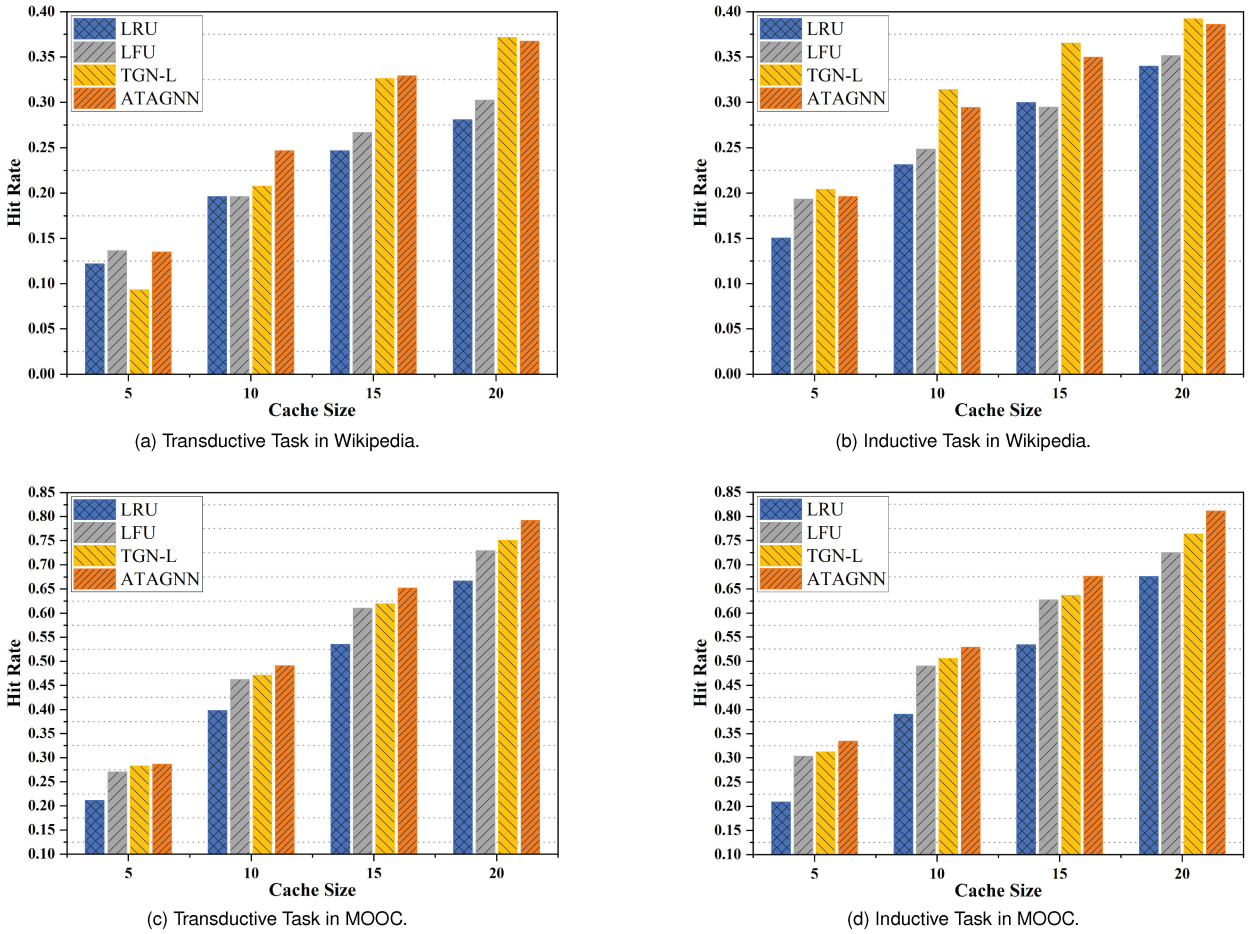
On the other hand, Fig. 8 displays the comparison between our model and TGN-L concerning the Wikipedia dataset with various update periods. The performance of both models gradually decreases with the increase of update period, and ATAGNN finally surpasses TGN-L at an update frequency of every hour. In other words, our model relies more on the history messages and an appropriate memory update is more essential and influential for our ATAGNN model when TGN-L also owns a similar prediction capability.

Moreover, we also examine the performance of MOOC dataset with different memory update periods (i.e., $T_u = 24,\ 1$). As shown in Table V, when the predictions are carried out based on the users that have requested content in the last hour, ATAGNN can always precede TGN-L. However, unlike the results in Wikipedia, the performance of MOOC with $T_u = 24$ is better than that with $T_u = 1$. On the other hand, if we have prior knowledge about the users, we can discover that the results have the same trend as Wikipedia and are



(a) Transductive Task (b) Inductive Task

Fig. 6. 24-hour hit rate performance of MOOC datasets with different algorithms in Transductive and inductive tasks.



(a) Transductive Task in Wikipedia. (b) Inductive Task in Wikipedia.

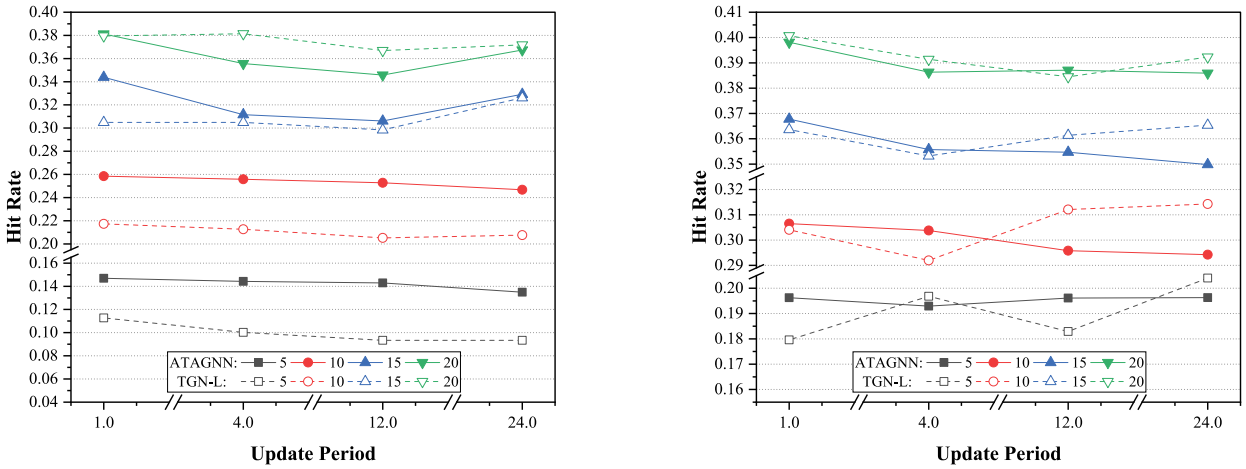
(c) Transductive Task in MOOC. (d) Inductive Task in MOOC.

Fig. 7. Hit rate comparison with different cache sizes in different datasets.

much better than the former setting. We also discover that the distribution of the number of users requesting content within two consecutive hours is more fluctuating in MOOC, which may explain the divergence of the results, especially when the prediction performance is also not as good as in Wikipedia.

VI. Conclusion

In this paper, we develop an AoI-based temporal attention graph neural network (ATAGNN) to maximize the precision of users' interest prediction in ICN. By aggregating the history interaction messages with self-attention mechanism, the model is able to generate a vector with rich temporal features. Furthermore, in order to tackle the problem of staleness, the concept of AoI is specifically introduced to exclude stale information for better refining history. The results based on two real-world datasets prove the superiority of the ATAGNN model over other neural network models. Because of its superior performance, a caching strategy totally based on the ATAGNN's prediction results with an appropriate memory



(a) Transductive Task. (b) Inductive Task.

Fig. 8. Hit rate comparison with different update periods in Wikipedia.

update period also wins a great improvement compared to the traditional algorithms or the best baseline based method. However, we also discover that the performance of our model still has room for further improvement, especially when the dataset is very sparse. Hence, we consider taking the semantic information of content into consideration while using the DGNN, for which semantic information may also imply the users' intentions and provide some implicit connections to help us better model the structural patterns of the graph.

In addition to the application of content caching, as we said before, with the development of intelligent-endogenesis in network, more and more communication services will rely on AI for processing, leading the pre-deployment of the AI model even more important. We believe the popularity prediction and deployment of AI models share many similarities with caching content, and look forward to extending the scenario to the pre-deployment of AI models.

APPENDIX
PROOF OF THE EFFECTIVENESS OF THE AGE FILTER

According to the results from TGN-L and TGN-M [32] in Table II, we can observe that the most recent message is more important on the final predictions than averaging all history features. On the one hand, the difference may derive from that averaging smooths historical features too much. On the other hand, the influence of a message may be inversely proportional to its age and some are too old to be positive. The results of Table V also prove this.

Theorem 1: We further assume that we can map the age of information to the effectiveness of the current moment with a function $\mathcal{P}(a_m)$, where a_m is the age of m-th history. Due to the self-attention mechanism, all the chosen messages' influence can be denoted as $\sum_{m=0}^{N} \beta_m \mathcal{P}(a_m)$, where β_m is the attention coefficient we obtain from Eq. (8) and N is the selected number. We believe when the proper threshold a_M satisfies $\mathcal{P}(a_M) \leq \frac{1}{Z} \sum_{m'=0}^{N'} \Delta_{\beta'_m} \mathcal{P}(a_{m'})$, it will filter useful historical messages and improve our final predictions.

Proof: If we want to filter the valuable information with its age adaptively, the total effectiveness of historical information turns into $\sum_{m'=0}^{N'} \beta'_{m'} \mathcal{P}(a_{m'})$. And $N \geq N' = M - 1$. If the final results improve, we have the following condition:

$$\sum_{m'=0}^{N'} \beta'_{m'} \mathcal{P}(a_{m'}) - \sum_{m=0}^{N} \beta_m \mathcal{P}(a_m) \geq 0 \tag{17}$$

And it can also be considered as:

$$\sum_{m'=0}^{N'} (\beta'_{m'} - \beta_{m'}) \mathcal{P}(a_{m'}) - \sum_{m=M}^{N} \beta_m \mathcal{P}(a_m) \geq 0$$
$$\sum_{m'=0}^{N'} (\beta'_{m'} - \beta_{m'}) \mathcal{P}(a_{m'}) \geq \sum_{m=M}^{N} \beta_m \mathcal{P}(a_m) \tag{18}$$

where a_M is the threshold that we need. To solve this inequality, we can relax the components on the right side:

$$\sum_{m'=0}^{N'} (\beta'_{m'} - \beta_{m'}) \mathcal{P}(a_{m'}) \geq (N - M + 1) \hat{\beta} \mathcal{P}(a_M)$$
$$\mathcal{P}(a_M) \leq \frac{1}{Z} \sum_{m'=0}^{N'} \Delta_{\beta'_m} \mathcal{P}(a_{m'}) \tag{19}$$

where $\hat{\beta}$ is the maximum value in a_m for $m \in (M, M+1, \ldots, N)$ and $Z = (N - M + 1)\hat{\beta}$, $\Delta_{\beta'_m} = \beta'_{m'} - \beta_{m'}$. Moreover, due to the negative correlation between AoI and its effectiveness, $\mathcal{P}(a_M)$ is also the maximum one. Obviously, if Eq. (19) is satisfied, the (17) will be satisfied as well. ■

Remark 1: In practice, we find that we can solve this inequality by using a two-layer MLP. We hope the first layer can achieve the component of the right side and the second layer can fit the inverse function of the mapping between age and effectiveness. On the other hand, Eq. (15) is adopted to make the backpropagation much easier.

REFERENCES

[1] "Cisco annual Internet report (2018–2023) white paper," Cisco, San Jose, CA, USA, White Paper, 2020. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.pdf

[2] Q. Li, W. Shi, X. Ge, and Z. Niu, "Cooperative edge caching in software-defined hyper-cellular networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2596–2605, Nov. 2017.
[3] H. Gu and H. Wang, "A distributed caching scheme using non-cooperative game for mobile edge networks," *IEEE Access*, vol. 8, pp. 142747–142757, 2020.
[4] M. Cha, H. Kwak, P. Rodriguez, Y.-Y. Ahn, and S. Moon, "I tube, you tube, everybody tubes: Analyzing the world's largest user generated content video system," in *Proc. SIGCOMM*, New York, NY, USA, 2007, pp. 1–14.
[5] S. Liao et al., "Cognitive popularity based AI service sharing for software-defined information-centric networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 7, no. 4, pp. 2126–2136, Oct.–Dec. 2020.
[6] P. Yang, N. Zhang, S. Zhang, L. Yu, J. Zhang, and X. Shen, "Content popularity prediction towards location-aware mobile edge caching," *IEEE Trans. Multimedia*, vol. 21, no. 4, pp. 915–929, Apr. 2019.
[7] G. S. Paschos, G. Iosifidis, M. Tao, D. Towsley, and G. Caire, "The role of caching in future communication systems and networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1111–1125, Jun. 2018.
[8] Z. Zhang, C.-H. Lung, M. St-Hilaire, and I. Lambadaris, "An SDN-based caching decision policy for video caching in information-centric networking," *IEEE Trans. Multimedia*, vol. 22, no. 4, pp. 1069–1083, Apr. 2020.
[9] S. Lederer, C. Mueller, C. Timmerer, and H. Hellwagner, "Adaptive multimedia streaming in information-centric networks," *IEEE Netw.*, vol. 28, no. 6, pp. 91–96, Nov./Dec. 2014.
[10] B. Ahlgren, C. Dannewitz, C. Imbrenda, D. Kutscher, and B. Ohlman, "A survey of information-centric networking," *IEEE Commun. Mag.*, vol. 50, no. 7, pp. 26–36, Jul. 2012.
[11] S. O. Somuyiwa, A. György, and D. Gündüz, "A reinforcement-learning approach to proactive caching in wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 6, pp. 1331–1344, Jun. 2018.
[12] Q. Chen, W. Wang, F. R. Yu, M. Tao, and Z. Zhang, "Content caching oriented popularity prediction: A weighted clustering approach," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 623–636, Jan. 2021.
[13] D. Lee et al., "LRFU: A spectrum of policies that subsumes the least recently used and least frequently used policies," *IEEE Trans. Comput.*, vol. 50, no. 12, pp. 1352–1361, Dec. 2001.
[14] A. Dan and D. Towsley, "An approximate analysis of the LRU and FIFO buffer replacement schemes," in *Proc. SIGMETRICS*, Boulder, CO, USA, May 1990, pp. 143–152.
[15] J. Zhou et al., "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, Jan. 2020.
[16] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," in *Proc. ICLR*, Vancouver, BC, Canada, Apr./May 2018, pp. 1–12.
[17] W. Fan et al., "Graph neural networks for social recommendation," in *Proc. WWW*, San Francisco, CA, USA, May 2019, pp. 417–426.
[18] D. Xu, C. Ruan, E. Korpeoglu, S. Kumar, and K. Achan, "Inductive representation learning on temporal graphs," in *Proc. ICLR*, Apr./May 2020, pp. 1–19.
[19] Y. Sun, E. Uysal-Biyikoglu, R. D. Yates, C. E. Koksal, and N. B. Shroff, "Update or wait: How to keep your data fresh," *IEEE Trans. Inf. Theory*, vol. 63, no. 11, pp. 7492–7508, Nov. 2017.
[20] Z. Ming, M. Xu, and D. Wang, "Age-based cooperative caching in information-centric networking," in *Proc. ICCCN*, Shanghai, China, Aug. 2014, pp. 1–8.
[21] Y. Wu, C. Wu, B. Li, L. Zhang, Z. Li, and F. C. M. Lau, "Scaling social media applications into geo-distributed clouds," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 689–702, Jun. 2015.
[22] Y. Zhang, X. Tan, and W. Li, "PPC: Popularity prediction caching in ICN," *IEEE Commun. Lett.*, vol. 22, no. 1, pp. 5–8, Jan. 2018.
[23] S. Mehrizi, A. Tsakmalis, S. Chatzinotas, and B. Ottersten, "A feature-based Bayesian method for content popularity prediction in edge-caching networks," in *Proc. WCNC*, Marrakech, Morocco, Apr. 2019, pp. 1–6.
[24] S. Lee, I. Yeom, and D. Kim, "T-caching: Enhancing feasibility of in-network caching in ICN," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 7, pp. 1486–1498, Jul. 2020.
[25] B. Hidasi, A. Karatzoglou, L. Baltrunas, and D. Tikk, "Session-based recommendations with recurrent neural networks," in *Proc. ICLR*, May 2016, pp. 1–10.
[26] S. Wu, Y. Tang, Y. Zhu, L. Wang, X. Xie, and T. Tan, "Session-based recommendation with graph neural networks," in *Proc. AAAI*, Honolulu, HI, USA, Jan./Feb. 2019, pp. 346–353.
[27] X. Wang, R. Wang, C. Shi, G. Song, and Q. Li, "Multi-component graph convolutional collaborative filtering," in *Proc. AAAI*, New York, NY, USA, Feb. 2020, pp. 6267–6274.
[28] A. Sankar, Y. Wu, L. Gou, W. Zhang, and H. Yang, "DySAT: Deep neural representation learning on dynamic graphs via self-attention networks," in *Proc. WSDM*, Houston, TX, USA, Feb. 2020, pp. 519–527.
[29] J. Skardinga, B. Gabrys, and K. Musial, "Foundations and modeling of dynamic networks using dynamic graph neural networks: A survey," *IEEE Access*, vol. 9, pp. 79143–79168, 2021.
[30] R. Trivedi, M. Farajtabar, P. Biswal, and H. Zha, "DyRep: Learning representations over dynamic graphs," in *Proc. ICLR*, New Orleans, LA, USA, May 2019, pp. 1–25.
[31] A. Vaswani et al., "Attention is all you need," in *Proc. NeurIPS*, Long Beach, CA, USA, Dec. 2017, pp. 6000–6010.
[32] E. Rossi, B. Chamberlain, F. Frasca, D. Eynard, F. Monti, and M. Bronstein, "Temporal graph networks for deep learning on dynamic graphs," in *Proc. ICML Workshop Graph Represent. Learn.*, Jul. 2020, pp. 1–9.
[33] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. NeurIPS*, Long Beach, CA, USA, Dec. 2017, pp. 1025–1035.
[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.
[35] K. Cho et al., "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. EMNLP*, Doha, Qatar, Oct. 2014, pp. 1724–1734.
[36] A. Kosta, N. Pappas, and V. Angelakis, "Age of information: A new concept, metric, and tool," *Found. Trends Netw.*, vol. 12, no. 3, pp. 162–259, 2017.

Jianhang Zhu (Graduate Student Member, IEEE) received the B.S. degree in communication engineering from Jilin University, Changchun, China, in June 2020. He is currently pursuing the Eng.D. degree with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou. His research interest includes graph neural network and edge computing.

Rongpeng Li (Member, IEEE) is currently an Associate Professor with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. He was a Research Engineer with the Wireless Communication Laboratory, Huawei Technologies Company Ltd., Shanghai, China, from August 2015 to September 2016. He was a Visiting Scholar with the Department of Computer Science and Technology, University of Cambridge, Cambridge, U.K., from February 2020 to August 2020. His research interest currently focuses on networked intelligence for communications evolving. He received the Wu Wenjun Artificial Intelligence Excellent Youth Award in 2021 and the sponsorship "the National Postdoctoral Program for Innovative Talents" in 2016. He serves as an Editor for *China Communications*.

Guoru Ding received the B.S. degree (Hons.) in electrical engineering from Xidian University, Xi'an, China, in 2008, and the Ph.D. degree (Hons.) in communications and information systems from the College of Communications Engineering, Nanjing, China, in 2014.

He is currently a Professor with the College of Communications Engineering, Nanjing. From 2015 to 2018, he was a Postdoctoral Research Associate with the National Mobile Communications Research Laboratory, Southeast University, Nanjing. His research interests include cognitive radio networks, massive MIMO, machine learning, and data analytics over wireless networks. He has received the Excellent Doctoral Thesis Award of the China Institute of Communications in 2016, the Alexander von Humboldt Fellowship in 2017, and the 14th IEEE COMSOC Aisa-Pacific Outstanding Young Researcher Award in 2019. He was a recipient of the Natural Science Foundation for Distinguished Young Scholars of Jiangsu Province, China, and six best paper awards from international conferences, such as the IEEE VTC-FALL 2014. He has served as a Guest Editor for the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS (Special Issue on Spectrum Sharing and Aggregation in Future Wireless Networks) and the *Chinese Journal of Aeronautics* (Special Issue on When Aeronautics Meets 6G and AI). He is currently an Associate Editor of the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING and an Editor of *China Communications*.

Chan Wang received the B.Eng. degree in information and communication engineering and the Ph.D. degree in microelectronics and solid state electronics from Zhejiang University in 2012 and 2017, respectively. Since October 2017, she has been working with Zhejiang University, where she is currently a Postdoctoral Fellow with the College of Information Science and Electronic Engineering. Her research interests include channel coding, cooperative communication, machine learning for wireless communications, and ad hoc networks.

Jianjun Wu is the Chief Researcher and the Director of the Future Network Laboratory, Huawei Technologies. He is leading the future network architecture design in Huawei.

Zhifeng Zhao (Member, IEEE) received the B.E. degree in computer science and the M.E. and Ph.D. degrees in communication and information systems from the PLA University of Science and Technology, Nanjing, China, in 1996, 1999, and 2002, respectively. From 2002 to 2004, he acted as a Postdoctoral Researcher with Zhejiang University, Hangzhou, China, where his researches were focused on multimedia next-generation networks and softswitch technology for energy efficiency. From 2005 to 2006, he acted as a Senior Researcher with the PLA University of Science and Technology, where he performed research and development on advanced energy-efficient wireless router, ad-hoc network simulator, and cognitive mesh networking test-bed. From 2006 to 2019, he was an Associate Professor with the College of Information Science and Electronic Engineering, Zhejiang University. He is currently with the Zhejiang Lab, Hangzhou, as the Chief Engineering Officer. His research areas include software defined networks, wireless network in 6G, computing networks, and collective intelligence. He is the Symposium Co-Chair of ChinaCom 2009 and 2010. He is the Technical Program Committee Co-Chair of the 10th IEEE International Symposium on Communication and Information Technology in 2010.

Honggang Zhang (Senior Member, IEEE) was an Honorary Visiting Professor with the University of York, York, U.K., and an International Chair Professor of excellence with the Université Européenne de Bretagne and Supélec, France. He is the Chief Managing Editor of Intelligent Computing, a Science Partner Journal, as well as a Professor with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China. He has coauthored and edited two books: *Cognitive Communications: Distributed Artificial Intelligence (DAI), Regulatory Policy & Economics, Implementation* (John Wiley & Sons) and *Green Communications: Theoretical Fundamentals, Algorithms and Applications* (CRC Press), respectively. His research interests include cognitive radio and networks, green communications, mobile computing, machine learning, artificial intelligence, and the Internet of Intelligence. He is a co-recipient of the 2021 IEEE Communications Society Outstanding Paper Award and the 2021 IEEE INTERNET OF THINGS JOURNAL Best Paper Award. He was the leading Guest Editor for the Special Issues on Green Communications of the *IEEE Communications Magazine*. He served as a Series Editor for the *IEEE Communications Magazine* (Green Communications and Computing Networks Series) from 2015 to 2018, and the Chair of the Technical Committee on Cognitive Networks of the IEEE Communications Society from 2011 to 2012. He is an Associate Editor-in-Chief of *China Communications*.